VERIFYING CYBER-PHYSICAL SYSTEMS

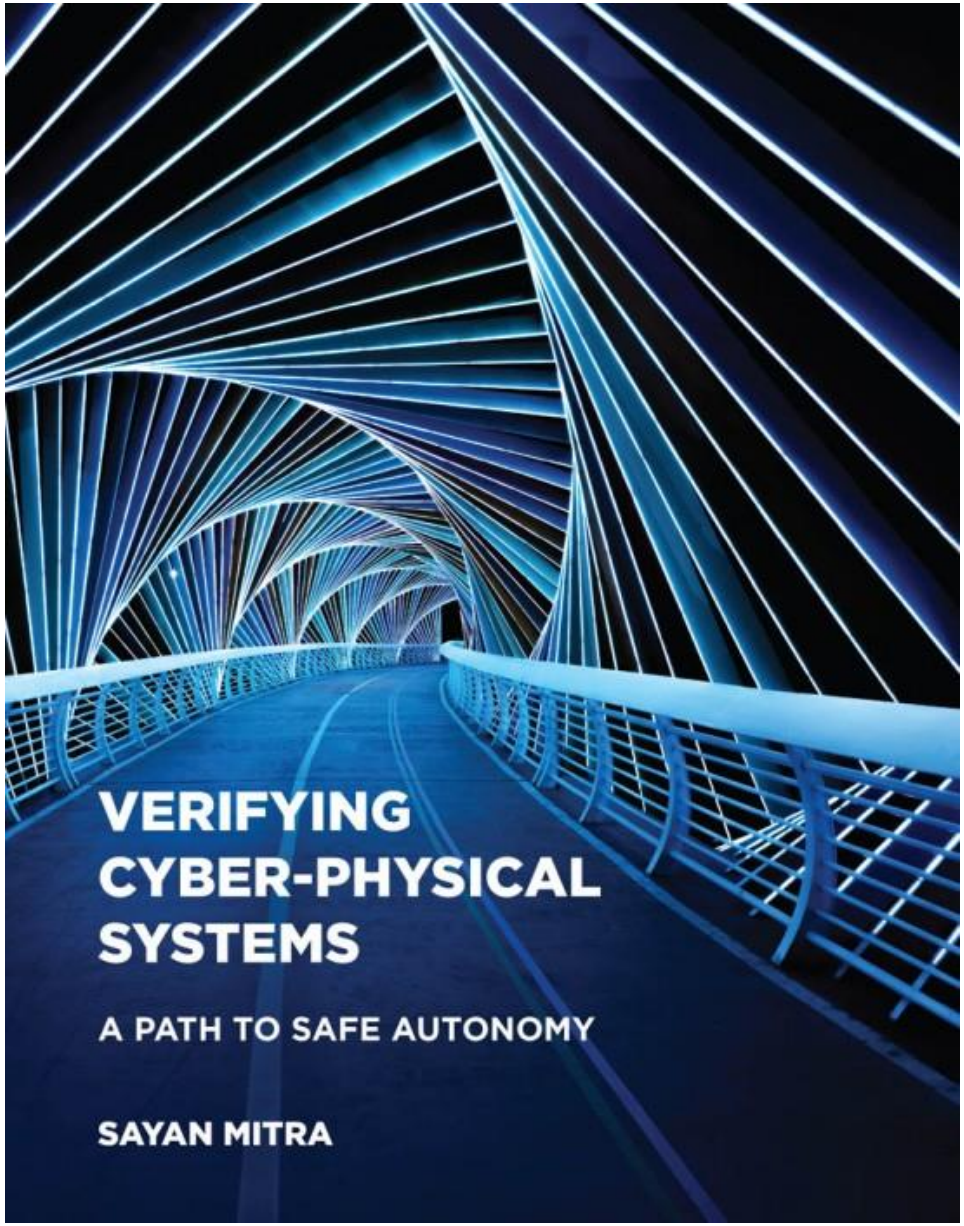
CHAPTER 4 - 5

Abenezer Taye
Graduate Research Assistant @ IASL

Date: Mar 23, 2022

CHAPTER 4:

MODELING CYBER-PHYSICAL SYSTEMS



INTRODUCTION

- **Cyber Physical System (CPS):** physical systems that are controlled by computer
- **Physical system modeling:** ordinary differential equations (ODE)
- **Computer operation:** Automata
- **CPS**: ODE + Automata (Hybrid Automata)

HYBRID AUTOMATA (HA)

- **Why study HA?**
  - When the two modeling paradigms collide → strange behavior occurs
    - ✓It is possible for a HA to perform infinitely many transitions in finite time
    - ✓It is possible for a HA to become unstable even though its ODEs are stable

- Example: Rimless wheel
- State change
  - **Instantaneously** → during impact & **Continuously** → over an interval of time

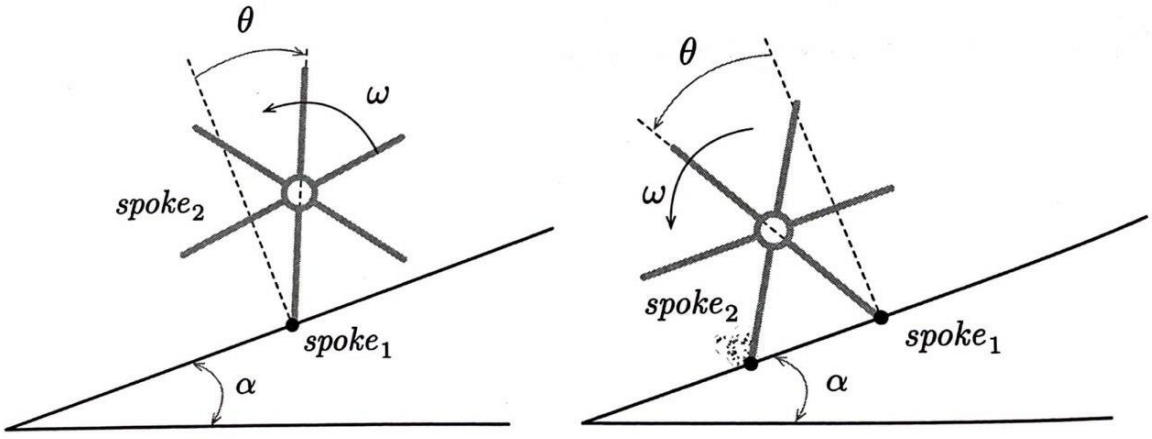


Figure 4.1
Motion of a rimless wheel down an inclined plane. Swinging around (*left*) and impact (*right*).

LANGUAGE FOR SPECIFYING HYBRID SYSTEMS

- **Actions:** lists the actions of the automaton (only **impact** in this case)
- **Variables:** lists the state variables with their initial conditions
- **Transitions:** describes how discrete variables change
- **Trajectories:** describes how continuous variables change

```
1  automaton RimlessWheel(α, μ : Real, n : Nat)
     const β : Real := 2π/n
3    type Spokes: enumeration [1,...,n]
     actions
5      impact

7    variables
       pivot:Spokes := 1
9      θ : Real := 0
       ω : Real := 0

     transitions                                  12
       impact
       pre θ ≥ β/2                                14
       eff pivot := pivot + 1  mod n
         θ := −β/2                                16
         ω := μω
                                                  18
     trajectories
       swing                                      20
       evolve
         d(θ) = ω                                 22
         d(ω) = sin(θ + α)
       invariant θ ≤ β/2                          24
```

Figure 4.2
HA model of RimlessWheel.

HYBRID AUTOMATA (HA)

- An HA $\mathcal{A}$ is a tuple $(V, \Theta, A, \mathcal{D}, \mathcal{T})$ where
  1. V is a set of variables called the state variables. The set $val(V)$ of valuations of V is the set of states
  2. $\Theta \subseteq val(V)$ is a nonempty set of start states
  3. A is a set of actions or transition labels
  4. $\mathcal{D} \subseteq val(V) \times A \times val(V)$ is called the set of transitions
  5. $\mathcal{T}$ is a set of trajectories for the variables in V

- An HA $\mathcal{A}$ is a deterministic HA
  - If the initial set Θ is a singleton set (a set with only value for one element) and
  - There is a single trajectory $\tau \in \mathcal{T}$ of non-zero duration with $\tau.fstate = v$

SPECIAL CLASSES OF HYBRID AUTOMATA (HA)

SWITCHED SYSTEMS

- It is an alternative formalism for describing cyber physical systems
  - Describes the system using a set of ODEs → one for each mode

$$\dot{x} = f_i(x), \qquad i \in [p], and\ p\ \in \mathbb{N}\ modes$$

$$\dot{x} = f_{\sigma(t)}(x), \qquad \sigma: \mathbb{R}_{\geq 0} \rightarrow [p]\ is\ swithing\ signal$$

```
automaton SwitchedSys(p : Nat, f : [p] × Real[n] → Real[n], σ : Seq[⟨Real, [p]⟩])

    actions
        switch(j : [p])

    variables
        x: Real[n]
        loc: [p]
        now: Real := 0
        index: Nat := 0

    transitions
        switch(j)
        pre σ(index) = ⟨now, j⟩ ∧ loc ≠ j
        eff loc := j
            index := index+1

    trajectories
        mode(i)
        evolve
            d(now) = 1
            d(x) = f(i, x)
        invariant loc = i ∧ now ≤ σ(index)[1]
```

Figure 4.3
HA specification of a generic switched system.

SPECIAL CLASSES OF HYBRID AUTOMATA (HA)

RECTANGULAR HYBRID AUTOMATA (RHA)

- A ***rectangular inclusion*** for a real-valued variable x is specified as

$$l_m \leq \frac{dx}{dt} \leq u_m$$

Where the constants $l_m, u_m \in \mathbb{R}$

- Rectangular HA is a HA in which
  1. The evolution of each continuous variable x in each mode m is specified by ***rectangular inclusion***
  2. The preconditions and the set of initial states are ***rectangles***
  3. The effects are either ***identity maps*** or ***assignments to rectangles***

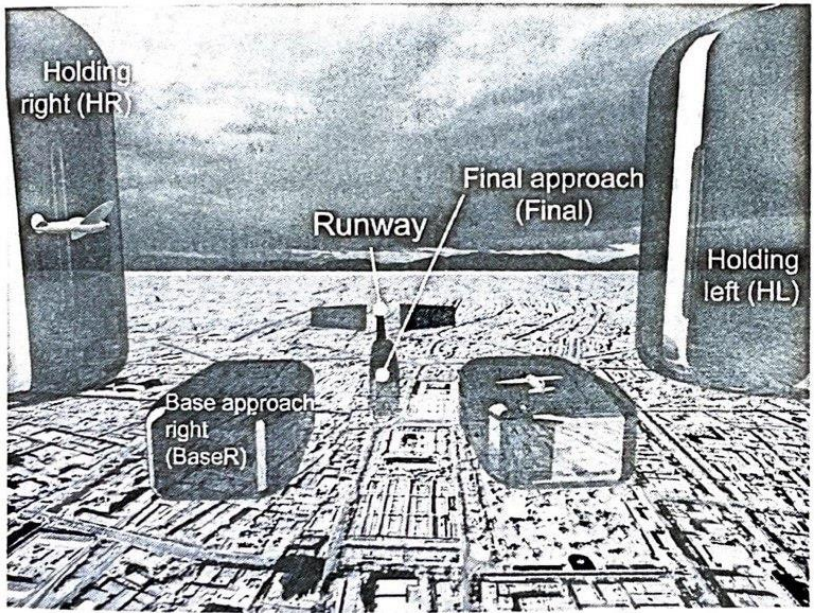
HYBRID AUTOMATA (HA)

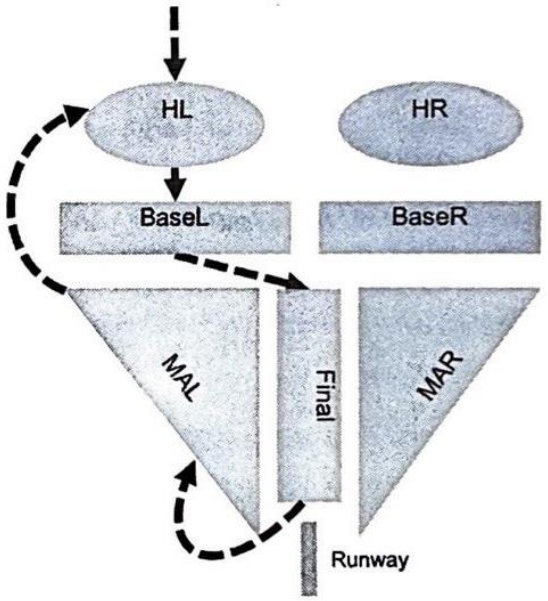
SEMANTICS FOR HYBRID SYSTEMS

- **Execution fragment:**
  - an alternating, possibly infinite sequence of actions and trajectories $\alpha = \tau_0 a_1 \tau_1 a_2 \ldots$, where each τ_i is a in $\mathcal{T}_{\mathcal{A}}$; and $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$
- $\boldsymbol{Reach(\Theta, T, k)}$**:** the set of states reachable from Θ in a maximum of k steps and in a maximum of T time
- ***Invariant (S):*** a set of states that contains $Reach(\Theta)$
- Stability (***globally uniformly asymptotically stable***)
  - if for any $\in > 0$ and any state v_0, there is a time $T_{\in,v_0}$ such that for any execution fragment α starting from v_0, for all $t \geq T_{\in,q_0}, |\alpha(t)| \leq \in$.

SMALL AIRCRAFT TRAFFIC MANAGEMENT SYSTEM

- Developed at NASA → to coordinate landing sequence of aircraft with light-weight ground infrastructure
- The abstraction captures the distributed coordination across aircraft and the time that it takes for aircraft to clear certain separation requirements
- Modeling assumption:
  - The airspace around the airport is divided into several holding and approach zones
  - The SATS protocol defines the conditions under which an aircraft participating in the protocol can move from one zone to the next zone
  - For easier analysis each aircraft → 1D point
  - Transitions → moving from one zone to another
  - Trajectories → motion in each zone



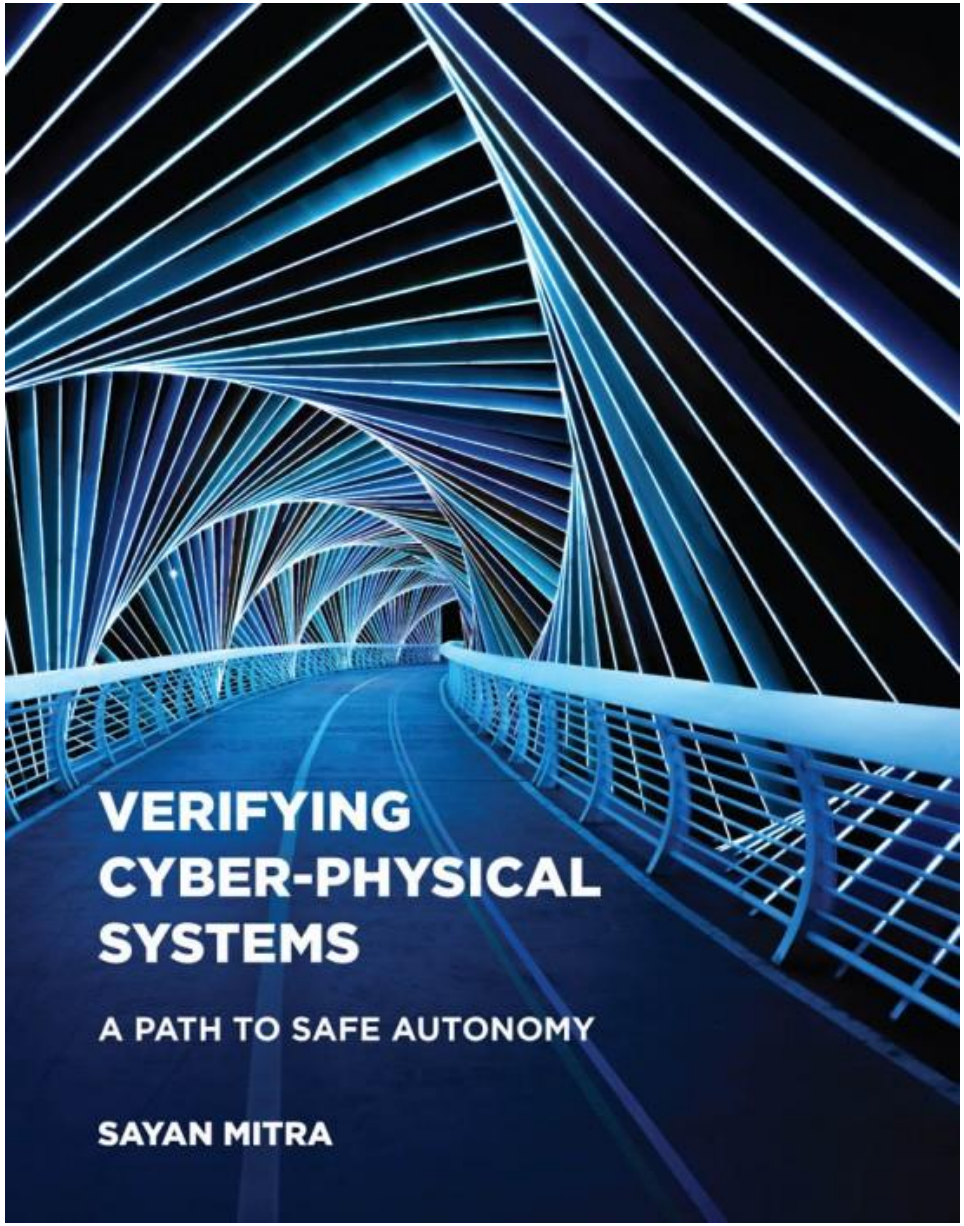


```
automaton SATS(n : Nat, L_Base, L_Safe, L_Final, v_min, v_max : Real)
  type Status enumeration
    [Fly,HL,BaseL, Final, MAL, Runway]
  type ID: enumeration [0,...,n]
  actions
    Fly2HL(i:ID), HL2BaseL(i:ID), BaseL2Final(i:ID),
    Land(i:ID), Miss(i:ID), MAL2HL(i:ID)

  variables
    status: [ID → Status] := Fly
    next: [ID → ID] := 0
    x: [ID → Real] := 0
    last: ID := 0

  transitions
    Fly2HL(i)
    pre status[i] = Fly ∧ i ≠ 0
    eff status[i] := HL
      next[i] := last
      last := i

    HL2BaseL(i)
    pre status[i] = HL ∧
      (next[i] = 0 ∨ status[next[i]] ≠ BaseL ∨
      (status[next[i]] = BaseL ∧ x[next[i]] ≥ L_Safe))
    eff status[i] := BaseL; x[i] = 0

    BaseL2Final(i)
    pre status[i] = BaseL ∧ x[i] ≥ L_Base
    eff status[i] := Final; x[i] := 0

    Land(i)
    pre status[i] := Final ∧ x[i] ≥ L_Final ∧ next[i] = 0
    eff status[i] := Runway
      if last = i then last := 0
      for j:ID
        if j ≠ i ∧ next[j] = i then next[j] := 0

    Miss(i)
    pre status[i] = BaseL ∨ status[i] = Final
    eff status[i] := MAL; x[i] := 0
      for j:ID
        if j ≠ i ∧ next[j] = i then next[j] := 0

    MAL2HL(i)
    pre status[i] := MAL ∧ x[i] ≥ L_MAL
    eff status[i] := HL; x[i] := 0
      if last ≠ i then next[i] := last
      last := i

  trajectories
    Traverse(i)
    evolve
      d(x[i]) ∈ [v_min, v_max]
    stop when
      (status[i] = BaseL ∧ x[i] ≥ L_Base) ∨
      (status[i] = Final ∧ x[i] ≥ L_Final) ∨
      (status[i] = MAL ∧ x[i] ≥ L_MAL)

    Hold(i)
    evolve
      d(x[i]) = 0
    invariant
      status[i] = Fly ∨ status[i] = HL ∨
      status[i] = Runway
```

Figure 4.12
Small Aircraft Transportation System.

CHAPTER 5:

COMPOSING MODELS



COMPOSING MODELS

COMPOSITION

- **Composition:** building large mathematical models from smaller modules
- ***Composing automata:*** creating a new automaton by composing two or more automaton
- ***Closure under composition:*** a property where bigger automaton A has the same type as A_1 and A_2 . (The formalism we use should achieve this property)
- We follow input/output automaton (IOA) formalism
  - Partitions its actions (and variables) into input, output, and internal actions
  - ***Input and output actions***: used for inter-automata synchronization and for state updates
  - ***Internal actions***: exclusively for state updates
  - ***Input enabledness***: a key assumption that ensures composition closure for IOA. It implies inputs can not be blocked.

COMPOSING MODELS

COMPOSING INPUT/OUTPUT AUTOMATA

- For each automaton $\mathrm{A}_i, i \in \{1,2\}$, the set of actions A_i is partitioned into three categories:
  - Internal actions (H_i): only bring about state changes of A_i and are not involved in any interactions
  - Output actions (O_i): are controlled by (i.e. triggered by) A_i and read by other automata
  - Input actions (I_i): are controlled by other automata and read by A_i
- An IOA $\mathcal{A} = < V, \Theta, A, D >$ is an automaton for which the set of actions A is partitioned into sets of input (I), output (O), and internal (H) actions. In addition, $\mathcal{A}$ satisfies the following input action-enabling condition: E1 (input action enabled).
- E1 ensures that $\mathcal{A}_i$ has well-defined transitions for every input action at any state.

COMPOSING MODELS

COMPATIBILITY AND COMPOSITION OF IOA

- A pair of IOA, $\mathcal{A}_1$ and $\mathcal{A}_2$, are compatible if for $i \neq j$
  1. $H_i \cap A_j = \emptyset$: internal actions of one automaton should not be triggered by any action of another automaton
  2. $O_i \cap O_j = \emptyset$: each output action should be controlled by only one automaton
  3. $V_i \cap V_j = \emptyset$: disjoint state space
- If $\mathcal{A}_1$and $\mathcal{A}_2$ are compatible, then their composition $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$ is defined to be $\mathcal{A} \triangleq < V, \Theta, A, \mathcal{D}, \mathcal{T} >$, where
  - $V = V_1 \cup V_2$
  - $\Theta = \{v \in val(V) \mid \forall i \in \{1,2\}, v \restriction V_i \in \Theta_i\}$
  - The overall set of actions $A = H \cup O \cup I = A_1 \cup A_2$, where
    - The set of internal actions $H = H_1 \cup H_2$
    - The set of output actions $O = O_1 \cup O_2$; and
    - The set of input actions $I = I_1 \cup I_2 \setminus O$
  - For each $v, v' \in val(V)$ and each $a \in A, v \xrightarrow{a} v'$

COMPOSING MODELS

COMPOSING HYBRID IOA

- An IOA $\mathcal{A} = < V, \Theta, A, D, \mathcal{T} >$ is a hybrid automaton for which
  1. The set of variables V is partitioned into sets of input (U), output (Y), and internal (X) variables. $val(X)$ is the state space
  2. $\Theta \subseteq val(X)$ is the set of start states
  3. The set of actions A is partitioned into sets of input (I), output (O), and internal (H) actions
  4. $\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of discrete transitions
  5. $\mathcal{T}$ is a prefix, suffix, and concatenated closed set of trajectories for V
  - In addition, $\mathcal{A}$ satisfies the following input-enabling conditions:
  1. ***E1 (input action enabled):*** At every state x, every input action $a \in I$ is enabled. That is, there is a post state x′ such that $x \xrightarrow{a} x'$
  2. ***E2 (input trajectory enabled):*** from every state x, the automaton should be able to react to any trajectory v of the input variables in U.

COMPOSING MODELS

COMPATIBILITY AND COMPOSITION OF HIOA

- If $\mathcal{A}_1$and $\mathcal{A}_2$ are compatible, then their composition $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$ is defined to be $\mathcal{A} \triangleq < V, \Theta, A, \mathcal{D}, \mathcal{T} >$, where

1. The overall set of variables $V = X \cup Y \cup U$, where
   - I. The set of internal variables $X = X_1 \cup X_2$
   - II. The set of output variables $Y = Y_1 \cup Y_2$; and
   - III. The set of input variables $U = U_1 \cup U_2 \backslash Y$
2. $\Theta = \{x \in val(X) \mid \forall i \in \{1,2\}, x \upharpoonright X_i \in \Theta_i\}$
3. The overall set of actions $A = H \cup O \cup I = A_1 \cup A_2$, where
   - I. The set of internal actions $H = H_1 \cup H_2$
   - II. The set of output actions $O = O_1 \cup O_2$; and
   - III. The set of input actions $I = I_1 \cup I_2 \backslash O$
4. For each x, $x' \in val(X)$ and each $a \in A, x \xrightarrow{a} x'$
5. A trajectory τ of V is in $\mathcal{T}$

ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS)

- More than 25% of reported traffic accidents are rear-end crashes
  - 85% of which happens on straight road
- Emergency breaking intervene automatically and prevent this type of accident
- Developed a HIOA to facilitate the testing of such system

Model parameters	State variables	Output variables
• Vehicle identifier $i \in ID$ • Its driver's reaction time (RT) • Initial separation from leading car (d_0) • Safety distance threshold (d_s) • The breaking profile (a)	• v_i and x_i position and velocity of vehicle • loc_i mode variable • $timer_i$ a stopwatch	• y_i → a copy of x_i • y_{i-1} → position of i's predecessor • 3 modes (breaking, cursing, and reacting) and 3 internal actions that trigger the transition among these modes

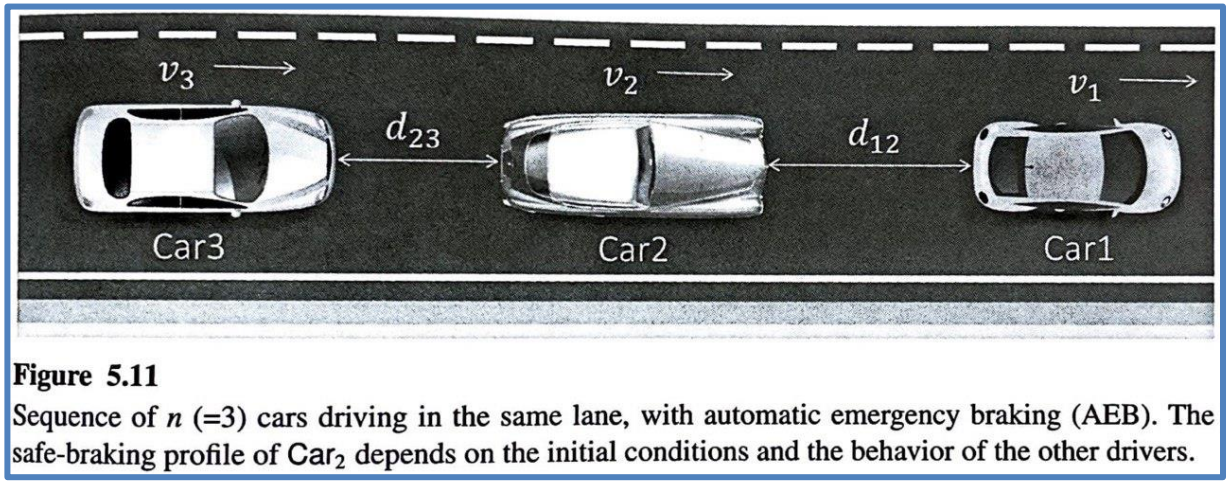


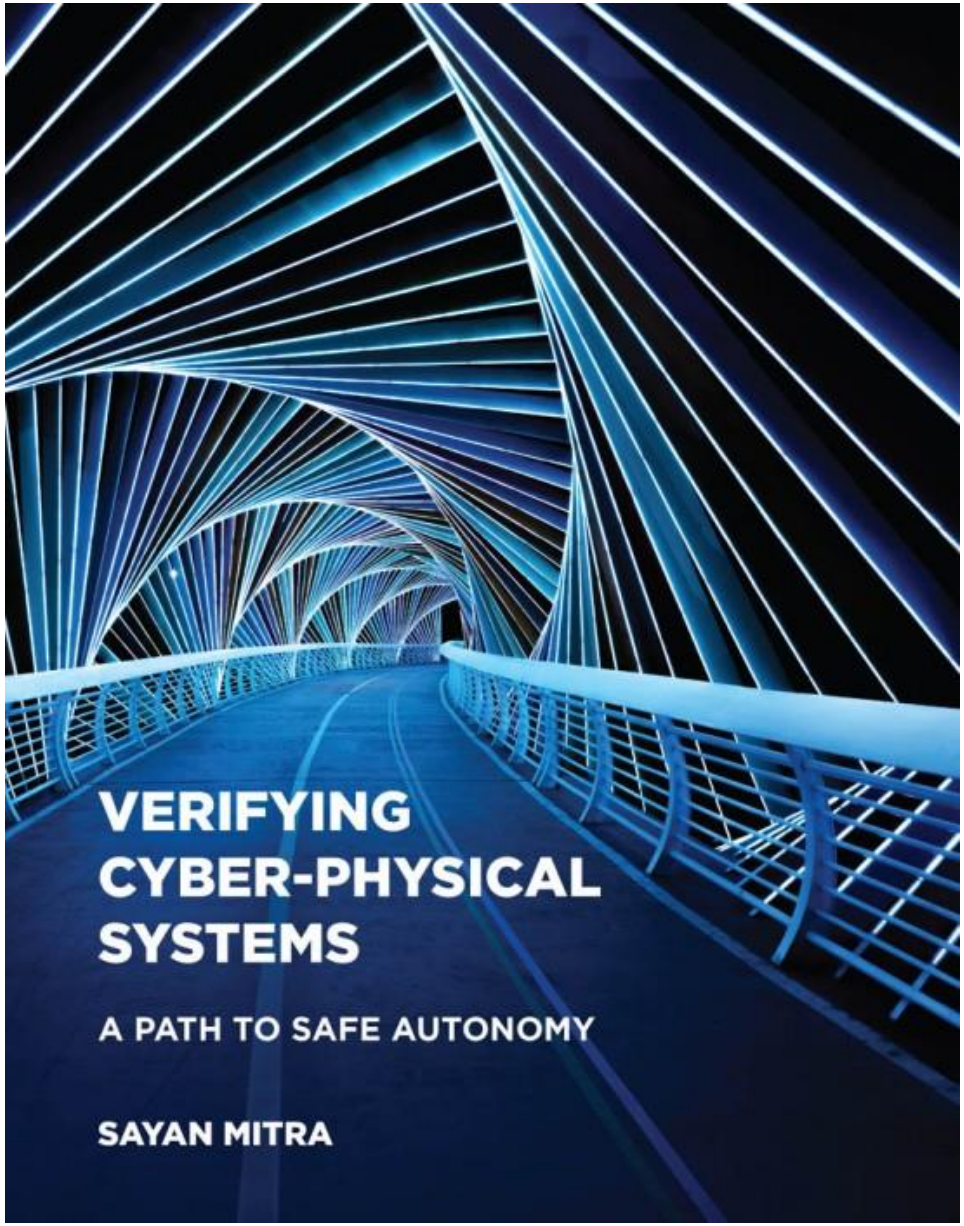
Figure 5.11
Sequence of n (=3) cars driving in the same lane, with automatic emergency braking (AEB). The safe-braking profile of Car_2 depends on the initial conditions and the behavior of the other drivers.

```
automaton SingleLaneCar(i : ID, RT, d_0, d_s : Real, a : [Real → Real])
  type Modes enumeration [braking, cruising, reacting]
  actions
    internal trigger_i, brake_i, cruise_i

  variables
    internal loc_i:Modes := cruising
    internal x_i := y_{i-1} + d_0
    internal v_i : Real
    internal timer_i : Real := 0
    input y_{i-1}
    output y_i

  transitions
    internal trigger_i
    pre loc_i = braking ∧ y_{i-1} − y_i ≤ d_s
    eff loc_i := reacting; timer_i := 0

    internal brake_i
    pre loc_i = reacting ∧ timer_i >= RT
    eff loc_i := braking

    internal cruise_i
    pre loc_i = braking ∧ y_{i-1} − y_i ≥ 2d_s
    eff loc_i := cruising

  trajectories
    CruiseReact
    evolve
      d(v_i) = 0
      d(x_i) = v_i
      d(timer_i) = 1
      y_i = x_i
    invariant (loc_i = cruising ∧ y_{i-1} − y_i ≥ d_s) ∨
      (loc_i = reacting ∧ timer_i ≤ q RT)

    braking
    evolve
      d(v_i) = a_i(v_i)
      d(x_i) = v_i
      d(timer_i) = 1
      y_i = x_i
    invariant
      (loc_i = braking ∧ y_{i-1} − y_i ≤ 2d_s)
```

Figure 5.12

Composition of HIOA modeling AEB on single-lane highways. The parameters d_{12} and d_{23} are set to be equal to d.

THANK YOU!!!


VERIFYING
CYBER-PHYSICAL
SYSTEMS
A PATH TO SAFE AUTONOMY
SAYAN MITRA